Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/MR/7440--04-8810

A Preliminary Investigation into the Estimation of River Depth Based on Meander Geometry

HILLARY C. MESICK
FRANK P. MCCREEDY

Mapping, Charting, and Geodesy Branch
Marine Geosciences Division

September 20, 2004

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
20-09-2004	Memorandum	

4. TITLE AND SUBTITLE

A Preliminary Investigation into the Estimation of River Depth Based on Meander Geometry

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Hillary C. Mesick and Frank P. McCreedy

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
74-7441-B4

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Marine Geoscience Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7440--04-8810

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Oceanographic Office
1002 Balch Blvd.
Stennis Space Center, MS 39529

10. SPONSOR / MONITOR'S ACRONYM(S)

NAVOCEANO

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

20041008 285

13. SUPPLEMENTARY NOTES

14. ABSTRACT

At times it becomes desirable to estimate the depths of rivers using aerial photography or other remotely sensed imagery. Due to water turbidity in many of the cases, optical techniques using water color or laser bathymetry are not feasible. This preliminary study investigates the potential use of meander geometry based on empirical equations as a possible method for the estimation of river depths for natural alluvial streams and rivers.

15. SUBJECT TERMS

Depths of rivers; Sensed imagery; Meander geometry; Empirical equations

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Hillary Mesick
Unclassified	Unclassified	Unclassified	UL	35	19b. TELEPHONE NUMBER *(include area code)* 228-688-5257

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Table of Contents

A PRELIMINARY INVESTIGATION INTO THE ESTIMATION OF RIVER DEPTH BASED ON MEANDER GEOMETRY

1.0 Introduction

At times it becomes desirable to estimate the depths of rivers using aerial photography or other remotely sensed imagery. Due to water turbidity in many of the cases, optical techniques using water color or laser bathymetry are not feasible. This preliminary study investigates the potential use of meander geometry based on empirical equations as a possible method for the estimation of river depths for natural alluvial streams and rivers.

2.0 Background

Early studies in fluvial geomorphology undertaken by Luna B. Leopold[1] and M.G. Wolman revealed statistically significant correlated relationships between meander radius of curvature, wavelength, width and river depth. These relationships were found to be consistent across a wide range of stream sizes; from small rivulets of glacial melt to the meander structure of the Gulf Stream. Subsequent studies by Garnett P. Williams[2] added additional information to the understanding of river meanders and channel size. Figure 1 illustrates a meander radius of curvature and wavelength.

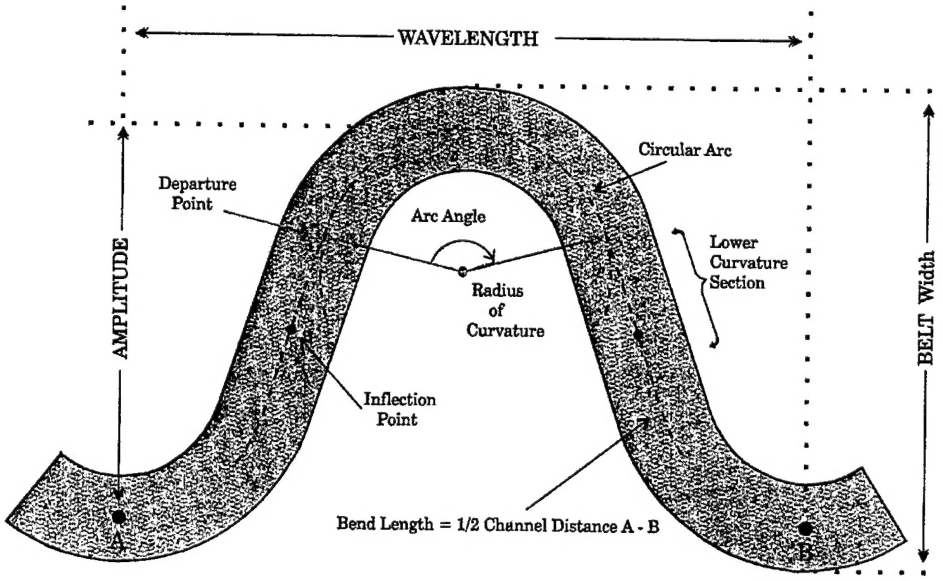

Figure *1. Schematic meander geometry (after Williams 1986)*

The following two figures taken from several sources illustrate some of the relationships among the various parameters:

[1] Leopold, L.B. and Wolman, M.G., 1960. River Meander. Bull. Geol. Soc. Am., 71:769-794
[2] Williams, G.P., 1986. River meanders and channel size. J. Hydrol., 88: 147-164
Manuscript approved July 30, 2004.

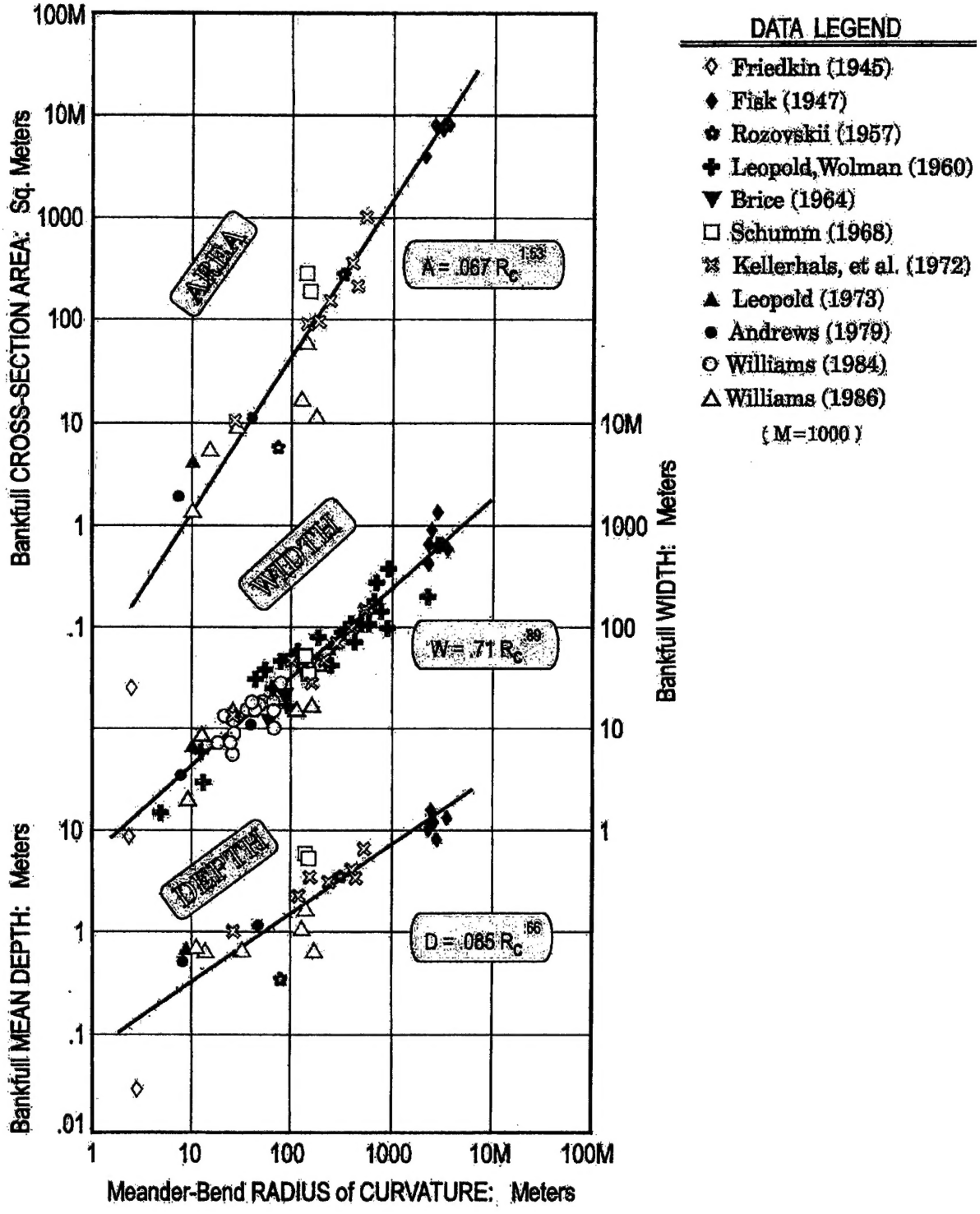

Figure 2
Relationships with Radius of Curvature (Rosgen, D., Applied River Morphology pg. 2-7, After Williams, Journal of Hydrology, 88: pg 157)

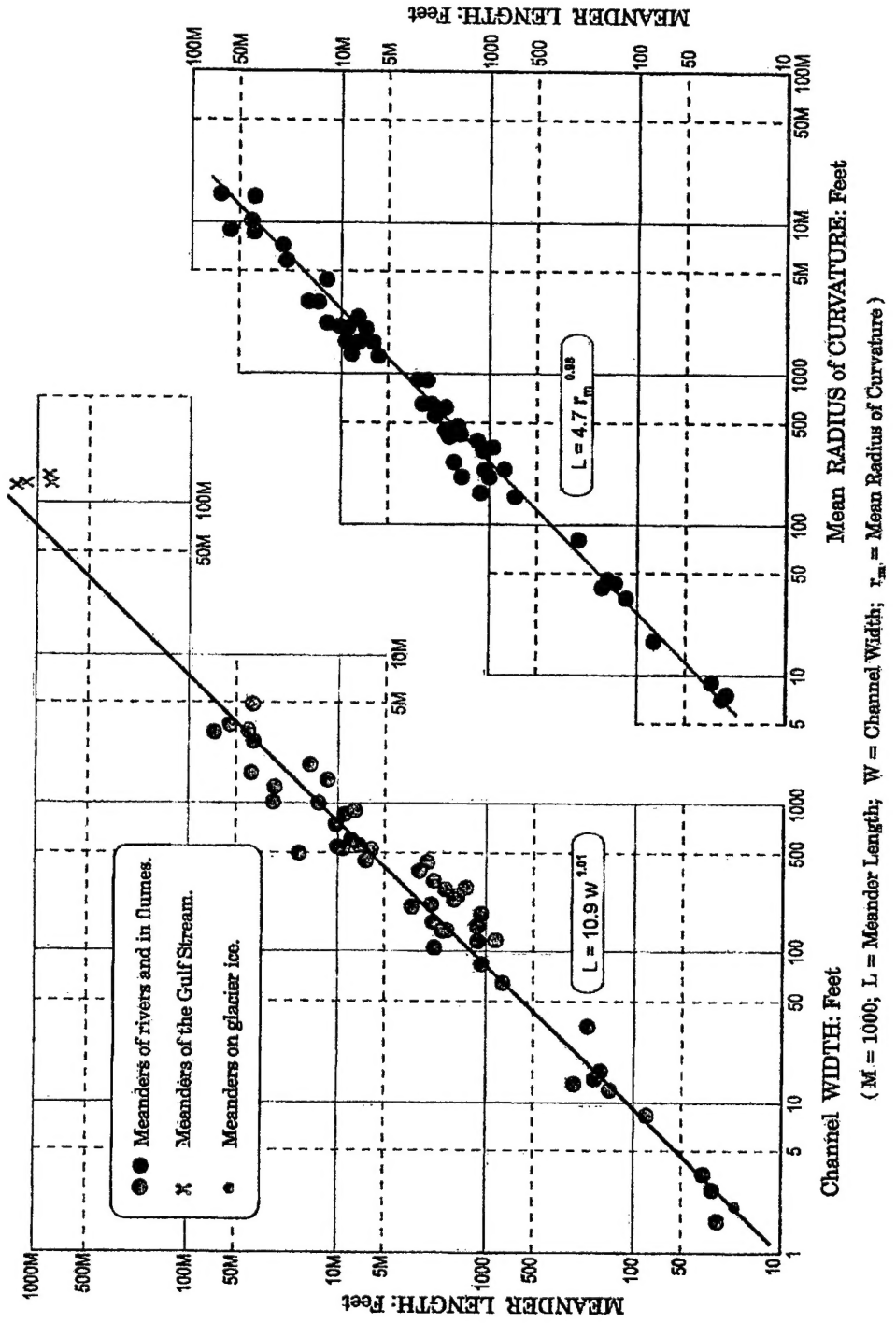

Figure 3

Relationships between Meander Wavelength, Channel Width and Radius of Curvature over a wide range of stream size (Rosgen, D., Applied River Morphology pg. 2-6; After Leopold 1964)

From these relationships Williams[3] derived several equations of best fit based on all the data available to him at the time, for various streams and rivers. Additionally an equation (eq.7) for Thalweg Depth from W. B. Leeder is also given. The equations of interest in the estimation of depth from meander geometry measurements are as follows:

$$D = 0.12\,W^{0.69} \qquad \text{eq 1.}$$

$$D = 0.09\,W^{0.59}\,K^{1.46} \qquad \text{eq 2.}$$

$$L_m = 240\,D^{1.52} \qquad \text{eq 3.}$$

$$R_c = 42\,D^{1.52} \qquad \text{eq 4.}$$

equations 3 and 4 can be rewritten to yield depth as a function of wavelength and radius of curvature

$$D = e^{(\ln(L_m/240)/1.52)} \qquad \text{eq 5.}$$

$$D = e^{(\ln(R_c/42)/1.52)} \qquad \text{eq 6.}$$

and for Thalweg Depth

$$W = 6.8\,D_t^{\,1.54} \qquad \text{eq 7.}$$

which can aslso be rewritten to yield

$$D_t = e^{(\ln(W/6.8)/1.54)} \qquad \text{eq 8.}$$

Where

D = depth mean hydraulic
D_t = depth thalweg
W = width
L_m = meander wavelenght
R_c = radius of curvature
K = sinuosity (ratio of stream channel distance to distance down slope)

---

[3] Williams, G.P., 1986. River meanders and channel size. J. Hydrol., 88: 147-164

To give the reader a feel for these functions, plots were generated and are shown in the following figures:

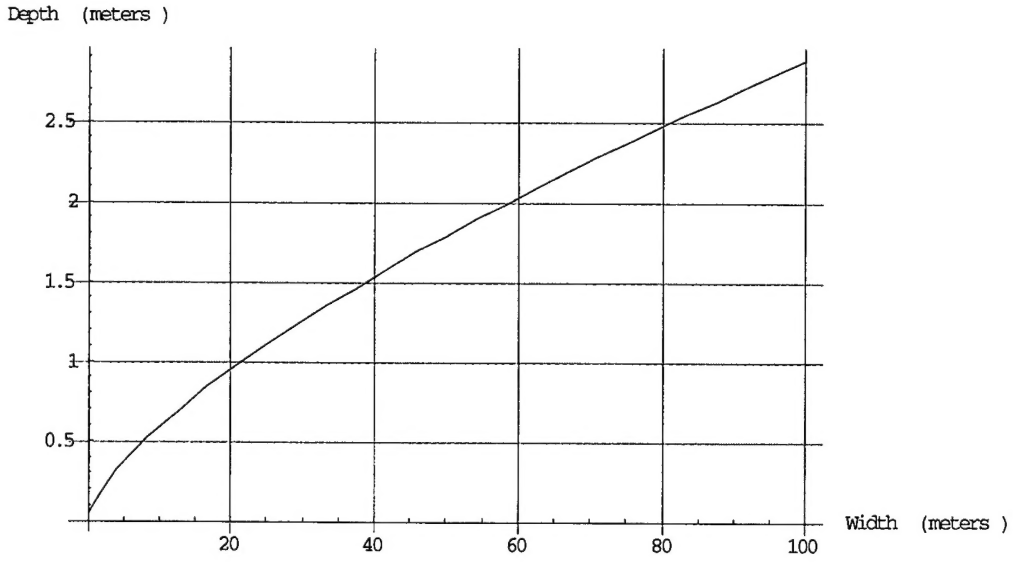

Figure 4.
Depth(meters) vs. Stream Width (0 – 100 meters)

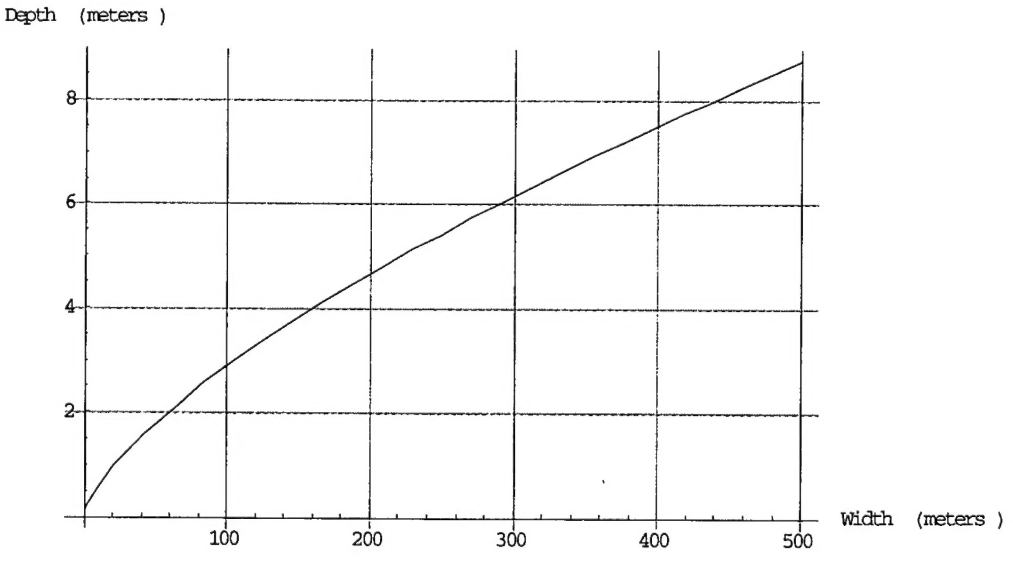

Figure 5.
Depth vs. Stream Width (0 – 500 meters)

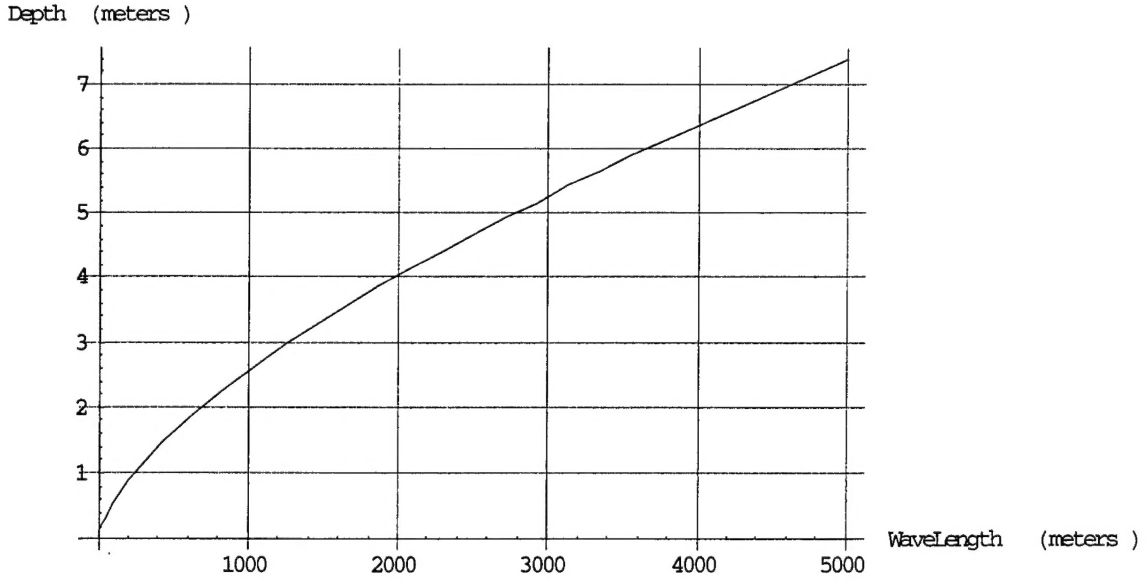

Figure 6.
Depth vs. Meander Wavelength

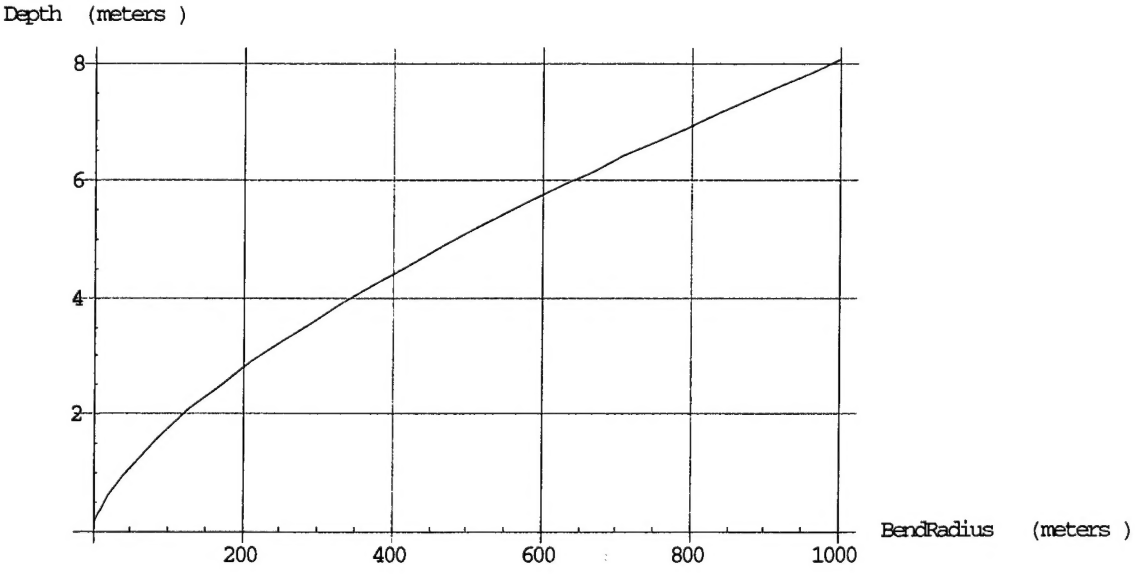

Figure 7.
Depth vs. Meander Bend Radius

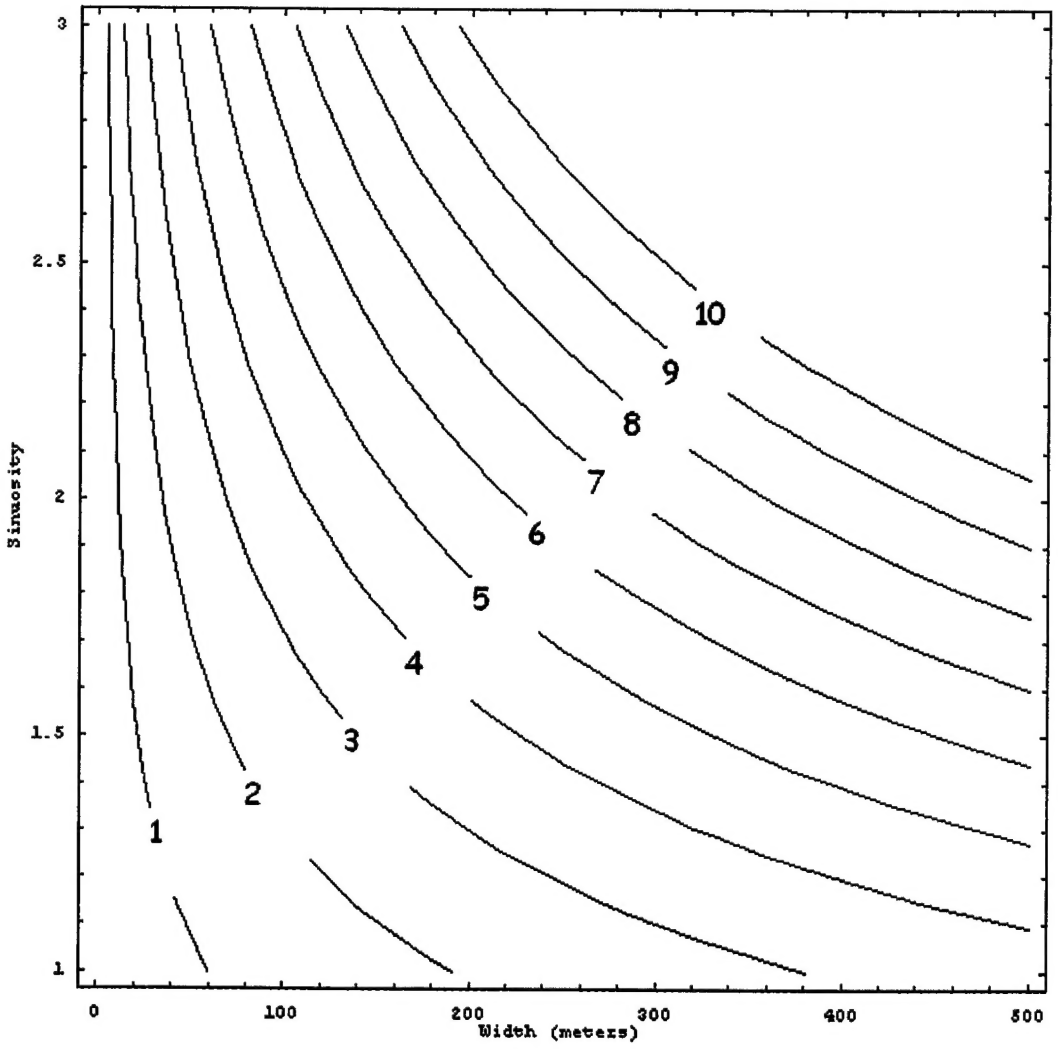

Figure 8.
Depth as a function of Width and Sinuosity (depth curves are numbered 1 – 10 meters)

In order to help the reader better understand the terms employed in the estimating equations the Figures 9 and 10 are provided.

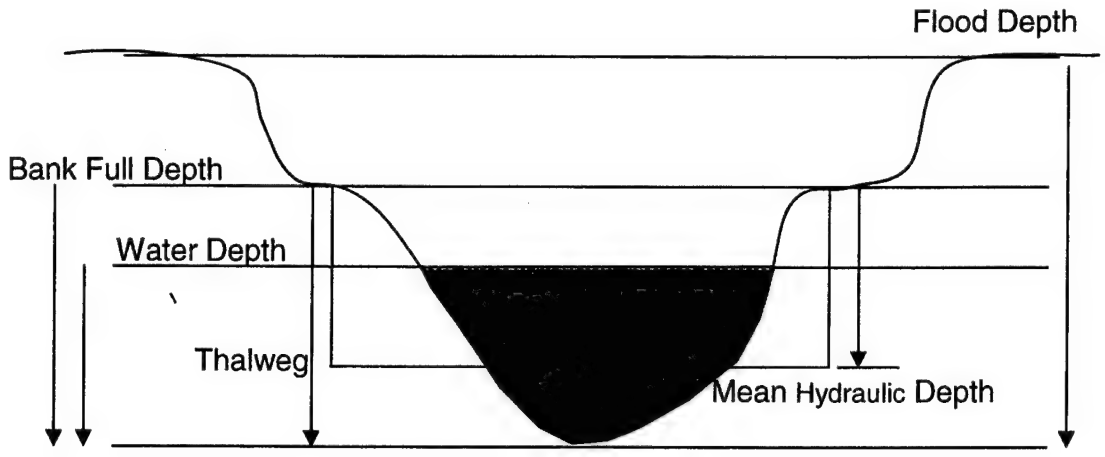

Figure 9.
Schematic Illustration of the different "depth" terms.

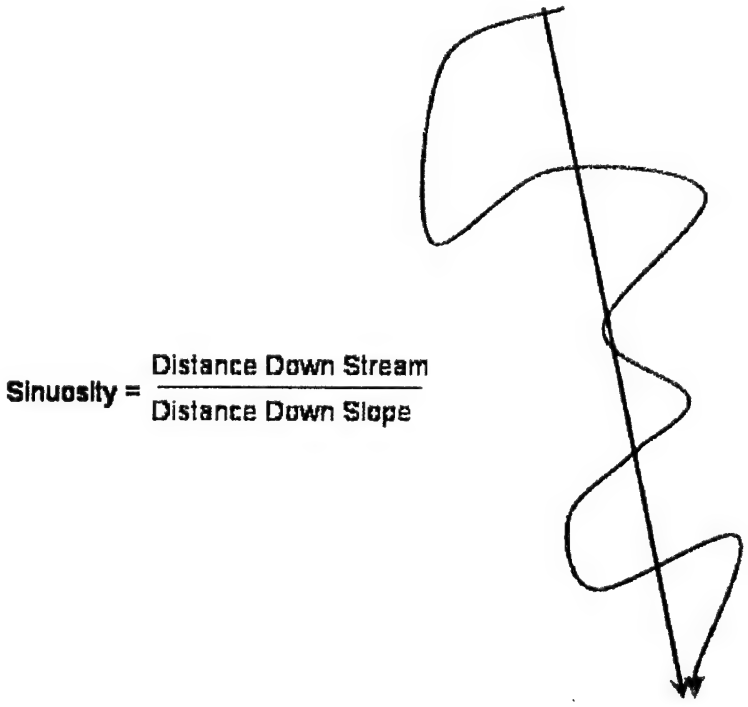

Figure 10.
Schematic Illustration of Sinuosity

3.0 Method

To locally verify these methods of depth estimation a section of the Wolf River located near the coast of southern Mississippi was selected. The region selected for testing was assumed to be sufficiently inland to be free of major tidal influence. The test area is shown in Figure 11.

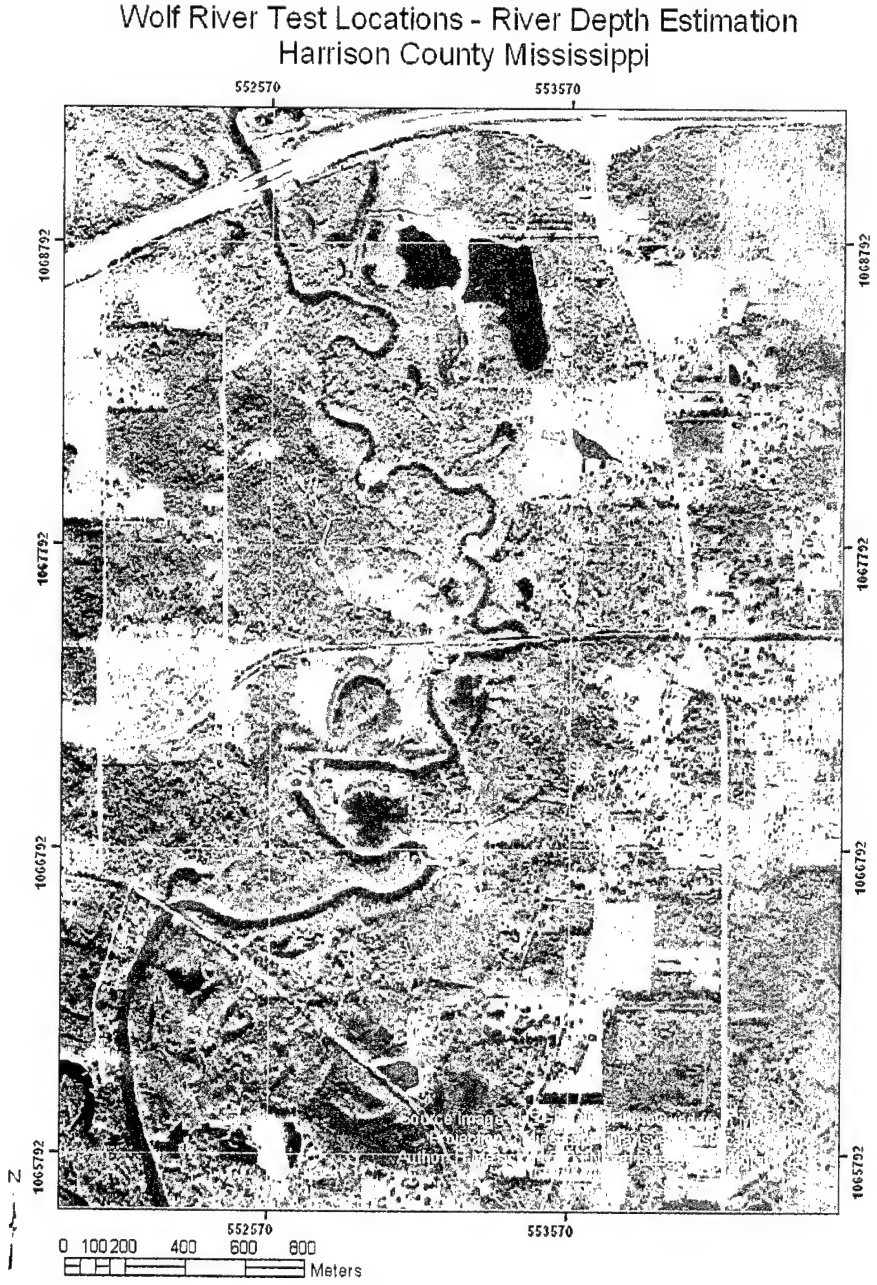

Figure 11.
Wolf River test locations.

USGS Ortho-Photo Quad imagery was obtained from MARIS, Mississippi Automated Resource Information System, in the Mississippi Transverse Mercator projection. All meander measurements were taken from this imagery using the ESRI ArcMap GIS software. For each of the five test locations, river width, bend radius, meander wavelength and sinuosity were measured. These measurements were entered into the appropriate equations and the results were recorded. To facilitate the calculations for repeated usage, a Java application program was written. This program is portable across a wide range of computer operating systems and is given in the appendix of this document. In addition to calculating a depth estimate based on each measured quantity, this program also computes an overall estimate based on an average of each of the individual calculations. Since the measurement of meander geometry is some what subjective, it was felt that an average of the four methods would produce more consistent results. These averaged values were used in the analysis. A screen capture of this application is shown in Figure 12.

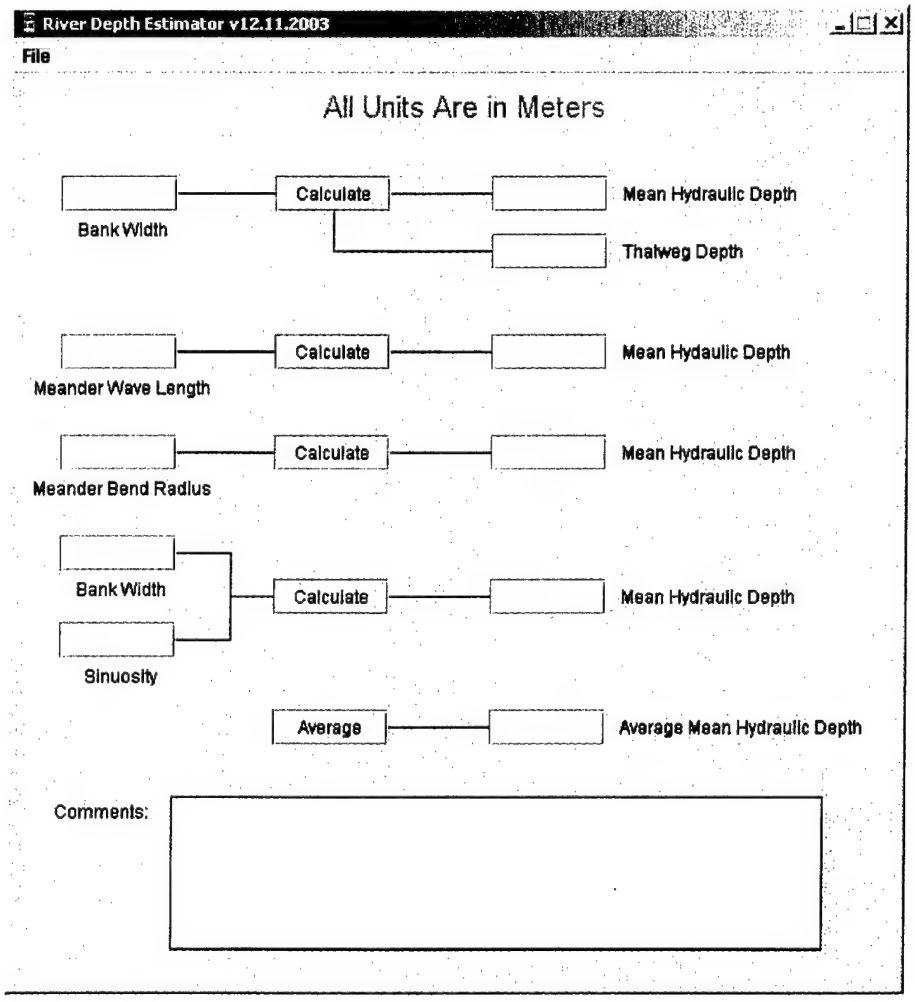

Figure 12. Screen Capture of the Java Application User Interface used for calculating average Mean Hydraulic Depth.

Field measurements were also made at each of the test locations. Depth cross-sections were measured using a lead-line for depth and "boat lengths" for horizontal displacement. Depth accuracy was estimated to be within +/- 10 cm. and horizontal distance accuracy to be within +/- 50 cm. The observed cross-sections were plotted and mean hydraulic depths were calculated from the results. An area for each cross section was calculated by trapezoidal integration, and from this area a mean depth derived that when multiplied by the measured width would yield an equivalent cross sectional area. (i.e. a rectangle of equal area). During the field measurements the thalweg depth[4] or greatest depth value was also determined. A composite of the cross-sections observed at the test locations is shown in Figure 13.

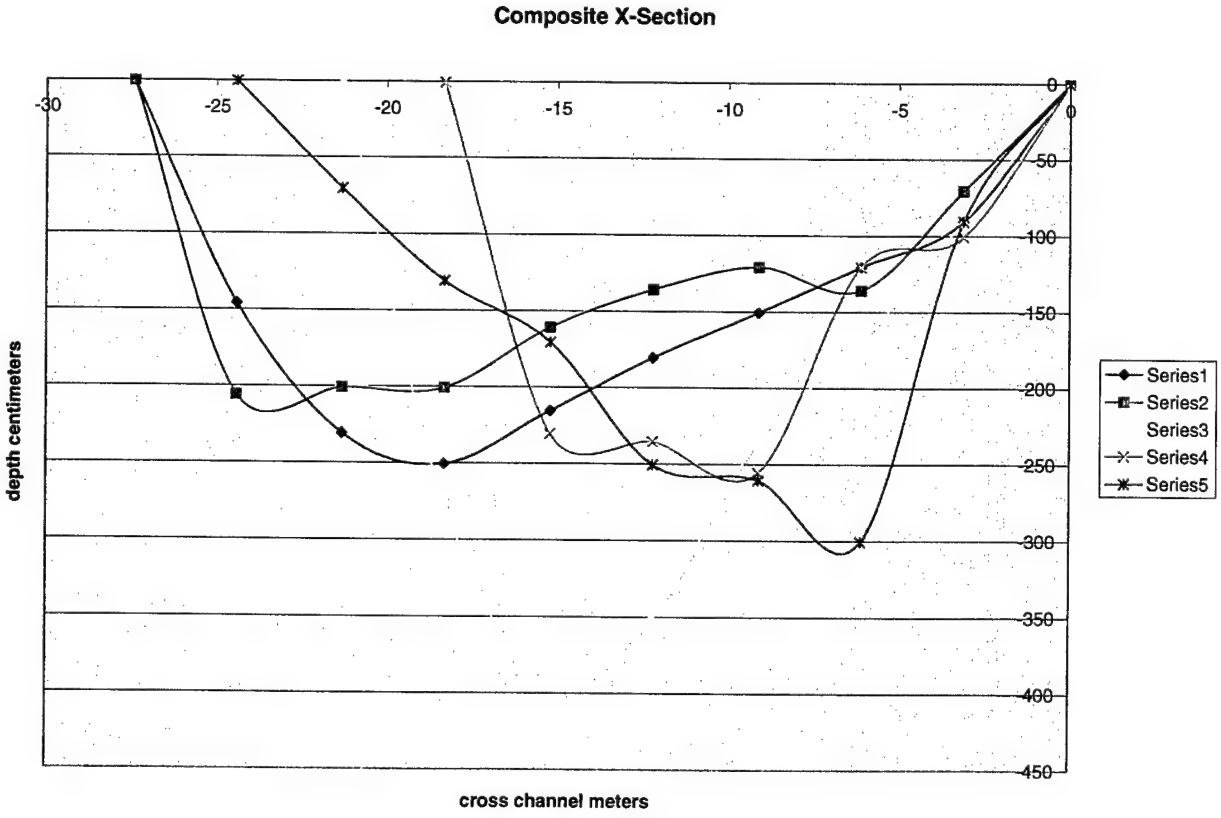

Figure 13.
Composite graph of the measured cross-sections at all five test locations.

[4] *thalweg is a physical geological term derived from two German words, "Thal" meaning valley, and "Weg" meaning path or trail, thus thalweg mean " the valley path", in this case the deepest part of the river channel.*

The following photograph provides a general idea of the environment and bank vegetation (Figure 14)

Figure 14.
Photograph Illustrating the general appearance and vegetation typical along the section of the Wolf River chosen for study.

4.0 Results

Table 1 shows the results for the five selected test sites. Width, Wavelength, Radius of Curvature, and Sinuosity as measured for each location are show along with the calculated and field measured values for Mean Hydraulic Depth, and Thalweg Depth. The estimated values in each case are based on an average of all four estimating equations, i.e. width, wavelength, radius of curvature and sinuosity at each cross-section location. The expected and measured results were in close agreement. The values associated with cross-section #3 were excluded in the overall error estimate. It is believe that the measured Thalweg Depth for this location was located behind a large submerged log, which caused excessive bottom scour thus making the measured value deeper than normal. The columns in Table 1, labeled Measured MHD and Measured Thalweg, and the two corresponding Error of Estimation columns each contain two sets of numbers. The first number is actual water depth (float a boat) and the second number is an adjusted depth for water level below the "Bank-full" elevation. It is the second number in this case that the estimating equations are designed to predict rather than the actual water depth observed. However the actual water depth is also given as the first number of the two.

Sec #	Width	Wave Length	Radius of Curvature	Sinuosity	Estimated MHD	Measured MHD	Error of Estimation	Estimated Thalweg	Measured Thalweg	Error of Estimation
1	40	402	112	2.1	1.8	1.6 // 1.9	0.2 // -0.1	3.1	2.5 // 2.8	0.6 // 0.3
2	28	425	139	2.6	1.9	1.3 // 1.6	0.6 // 0.3	2.4	2 // 2.3	0.4 // .1
3	24	250	52	2.4	1.4	2.2 // 2.5	-0.8 **	2.2	3.6 // 3.9	-1.4 **
4	31	251	55	2.5	1.5	1.6 // 1.9	-0.1 // -0.3	2.6	2.5 // 2.8	0.1 // -0.2
5	37	448	55	2.9	1.9	1.6 // 1.9	0.3 // 0	2.9	3 // 3.3	-0.1 // -0.4
All units are in Meters										
Bank Full Offset of 0.3 meters				Mean error			0.25// 0.03			.25// -0.05
				RMS			0.35 //**0.22**			0.37//**0.27**
** values not used										

Table 1, Tabulated Results

5.0 Conclusions

The method employed provided a reasonable estimate of the river depths in the areas selected. **Overall, the Root Mean Squared Error was 0.22 meters, which is within 12 percent of the River Mean Hydraulic Depth in the test area and an RMS of 0.27 meters for the thalweg, which is within 10 percent of the measured depth.**

Additional studies are needed to further validate this method for a wider range of river types. It should also be pointed out that actual water depths encountered in the field may vary widely from these estimates as a result of climatic trends or near term rain fall amounts. However, the estimating techniques are designed to determine the "bank full" depth and techniques may be developed to adjust these estimated values based on a comparison of currently observed "river width" to that of the "bank width".

6.0 Bibliography

Leopold, Luna B.; *A View of the River,* Harvard University Press, 1994.

Leopold, Luna B.; Wolman, M. Gordon; Miller, John P.; *Fluvial Processes in Geomorphology*; Dover 1964, 1992.

Leopold, Luna B., Langbein, W.B.; River Meanders; *Scientific American*, June 1966, pg 60-70.

Knighton, David; *Fluvial Forms & Processes, Arnold Press, 1998.*

Rosgen, Dave; *Applied River Morphology;* Wildland Hydrology, 1996.

Williams, Garnett P.; River Meanders and Channel Size*; Journal of Hydrology, 88 (1986)pg. 147-164.*

7.0 Acknowledgments

The authors would like to acknowledge and thank Mr. John Easton and Mr. John Daniel of the Naval Oceanographic Office, WSC for their support of this work

8.0 Appendix

This appendix contains the Java code used to implement the depth estimation equations.

```
package mil.navy.nrlssc.dmap.riverdepth;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import java.text.*;
import java.awt.print.*;
import javax.print.*;
import javax.print.attribute.*;
import javax.print.attribute.standard.*;
import java.io.*;
import java.awt.image.*;
import javax.imageio.*;

public class RiverDepthFrame extends JFrame implements ActionListener, Printable{
  JMenuBar jmbMenu;
  JMenu jmFile;
  JMenuItem jmiPrint, jmiSaveAs, jmiExit;

  JLabel jlblTitle2,
      jlblBankWidth1, jlblMeanHydraulicDepth1, jlblThalwegDepth,
jlblMeanderWaveLength, jlblMeanHydraulicDepth2,
      jlblMeanderBendRadius, jlblMeanHydraulicDepth3, jlblBankWidth2,
jlblSinuosity, jlblMeanHydraulicDepth4, jlblAverageMeanHydraulicDepth,
      jlblDepthRangeForAverageFlowConditions1, jlblComments;

  JTextField jtfBankWidth1, jtfMeanHydraulicDepth1, jtfThalwegDepth,
jtfMeanderWaveLength, jtfMeanHydraulicDepth2,
        jtfMeanderBendRadius, jtfMeanHydraulicDepth3, jtfBankWidth2, jtfSinuosity,
jtfMeanHydraulicDepth4, jtfAverageMeanHydraulicDepth;

  JTextArea jtaComments;

  JButton jbtnCalculate1, jbtnCalculate2, jbtnCalculate3, jbtnCalculate4, jbtnAverage;

  DecimalFormat oneFormatter;
  RiverDepthFrame frameReference;

  PrintService selectedPrinter;
  String strVersion;
  JLabel jlblPrintingVersion;
  String strSaveReportAsImageFolder;

  public RiverDepthFrame(){
```

```
        frameReference = this;
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(620, 680);
        setLocation( (int)(Toolkit.getDefaultToolkit().getScreenSize().width/2.0 -
getSize().width/2.0 + 0.5),
                (int)(Toolkit.getDefaultToolkit().getScreenSize().height/2.0 -
getSize().height/2.0 + 0.5) );
        setResizable(false);
        strVersion = "v12.12.2003";
        setTitle("River Depth Estimator " + strVersion);

setIconImage(Utility.getImageFromJar("mil/navy/nrlssc/dmap/riverdepth/icons/Calculato
r.gif"));
        getContentPane().setLayout(null);

        UIManager.put("Label.font", new Font(
((Font)(UIManager.get("Label.font"))).getName(), Font.PLAIN, 12));
        UIManager.put("Button.font", new Font(
((Font)(UIManager.get("Button.font"))).getName(), Font.PLAIN, 12));

        jmbMenu = new JMenuBar();

        jmFile = new JMenu("File");
        jmbMenu.add(jmFile);

        jmiPrint = new JMenuItem("Print");
        jmiPrint.addActionListener(this);
        jmFile.add(jmiPrint);

        jmiSaveAs = new JMenuItem("Save As...");
        jmiSaveAs.addActionListener(this);
        jmFile.add(jmiSaveAs);

        jmiExit = new JMenuItem("Exit");
        jmiExit.addActionListener(this);
        jmFile.add(jmiExit);

        setJMenuBar(jmbMenu);

        jlblTitle2 = new JLabel("All Units Are in Meters", SwingConstants.CENTER);
        jlblTitle2.setFont( new Font( ((Font)(UIManager.get("Label.font"))).getName(),
Font.PLAIN, 20));
        jlblTitle2.setSize(getSize().width - 20, 25);
        jlblTitle2.setLocation(10, 10);
        jlblTitle2.setForeground(Color.red);
        getContentPane().add(jlblTitle2);
```

```
        jtfBankWidth1 = new JTextField();
        jtfBankWidth1.setSize(80, 25);
        jtfBankWidth1.setLocation(33, 70);
        getContentPane().add(jtfBankWidth1);

        jlblBankWidth1 = new JLabel("Bank Width", SwingConstants.CENTER);
        jlblBankWidth1.setSize(130, 25);
        jlblBankWidth1.setLocation(10, 95);
        getContentPane().add(jlblBankWidth1);

        jbtnCalculate1 = new JButton("Calculate");
        jbtnCalculate1.setSize(80, 25);
        jbtnCalculate1.setLocation(180, 70);
        jbtnCalculate1.setMargin(new Insets(0, 0, 0, 0));
        jbtnCalculate1.addActionListener(this);
        getContentPane().add(jbtnCalculate1);

        jtfMeanHydraulicDepth1 = new JTextField();
        jtfMeanHydraulicDepth1.setSize(80, 25);
        jtfMeanHydraulicDepth1.setLocation(330, 70);
        getContentPane().add(jtfMeanHydraulicDepth1);

        jlblMeanHydraulicDepth1 = new JLabel("Mean Hydraulic Depth");
        jlblMeanHydraulicDepth1.setSize(180, 25);
        jlblMeanHydraulicDepth1.setLocation(420, 70);
        getContentPane().add(jlblMeanHydraulicDepth1);

        jtfThalwegDepth = new JTextField();
        jtfThalwegDepth.setSize(80, 25);
        jtfThalwegDepth.setLocation(330, 110);
        getContentPane().add(jtfThalwegDepth);

        jlblThalwegDepth = new JLabel("Thalweg Depth");
        jlblThalwegDepth.setSize(180, 25);
        jlblThalwegDepth.setLocation(420, 110);
        getContentPane().add(jlblThalwegDepth);

        jtfMeanderWaveLength = new JTextField();
        jtfMeanderWaveLength.setSize(80, 25);
        jtfMeanderWaveLength.setLocation(33, 180);
        getContentPane().add(jtfMeanderWaveLength);

        jlblMeanderWaveLength = new JLabel("Meander Wave Length",
    SwingConstants.CENTER);
        jlblMeanderWaveLength.setSize(130, 25);
```

```
        jlblMeanderWaveLength.setLocation(10, 205);
        getContentPane().add(jlblMeanderWaveLength);

        jbtnCalculate2 = new JButton("Calculate");
        jbtnCalculate2.setSize(80, 25);
        jbtnCalculate2.setLocation(180, 180);
        jbtnCalculate2.setMargin(new Insets(0, 0, 0, 0));
        jbtnCalculate2.addActionListener(this);
        getContentPane().add(jbtnCalculate2);

        jtfMeanHydraulicDepth2 = new JTextField();
        jtfMeanHydraulicDepth2.setSize(80, 25);
        jtfMeanHydraulicDepth2.setLocation(330, 180);
        getContentPane().add(jtfMeanHydraulicDepth2);

        jlblMeanHydraulicDepth2 = new JLabel("Mean Hydaulic Depth");
        jlblMeanHydraulicDepth2.setSize(180, 25);
        jlblMeanHydraulicDepth2.setLocation(420, 180);
        getContentPane().add(jlblMeanHydraulicDepth2);

        jtfMeanderBendRadius = new JTextField();
        jtfMeanderBendRadius.setSize(80, 25);
        jtfMeanderBendRadius.setLocation(33, 250);
        getContentPane().add(jtfMeanderBendRadius);

        jlblMeanderBendRadius = new JLabel("Meander Bend Radius",
    SwingConstants.CENTER);
        jlblMeanderBendRadius.setSize(130, 25);
        jlblMeanderBendRadius.setLocation(10, 275);
        getContentPane().add(jlblMeanderBendRadius);

        jbtnCalculate3 = new JButton("Calculate");
        jbtnCalculate3.setSize(80, 25);
        jbtnCalculate3.setLocation(180, 250);
        jbtnCalculate3.setMargin(new Insets(0, 0, 0, 0));
        jbtnCalculate3.addActionListener(this);
        getContentPane().add(jbtnCalculate3);

        jtfMeanHydraulicDepth3 = new JTextField();
        jtfMeanHydraulicDepth3.setSize(80, 25);
        jtfMeanHydraulicDepth3.setLocation(330, 250);
        getContentPane().add(jtfMeanHydraulicDepth3);

        jlblMeanHydraulicDepth3 = new JLabel("Mean Hydraulic Depth");
        jlblMeanHydraulicDepth3.setSize(150, 25);
        jlblMeanHydraulicDepth3.setLocation(420, 250);
```

```
getContentPane().add(jlblMeanHydraulicDepth3);

jtfBankWidth2 = new JTextField();
jtfBankWidth2.setSize(80, 25);
jtfBankWidth2.setLocation(33, 320);
getContentPane().add(jtfBankWidth2);

jlblBankWidth2 = new JLabel("Bank Width", SwingConstants.CENTER);
jlblBankWidth2.setSize(130, 25);
jlblBankWidth2.setLocation(10, 345);
getContentPane().add(jlblBankWidth2);

jtfSinuosity = new JTextField();
jtfSinuosity.setSize(80, 25);
jtfSinuosity.setLocation(33, 380);
getContentPane().add(jtfSinuosity);

jlblSinuosity = new JLabel("Sinuosity", SwingConstants.CENTER);
jlblSinuosity.setSize(130, 25);
jlblSinuosity.setLocation(10, 405);
getContentPane().add(jlblSinuosity);

jbtnCalculate4 = new JButton("Calculate");
jbtnCalculate4.setSize(80, 25);
jbtnCalculate4.setLocation(180, 350);
jbtnCalculate4.setMargin(new Insets(0, 0, 0, 0));
jbtnCalculate4.addActionListener(this);
getContentPane().add(jbtnCalculate4);

jtfMeanHydraulicDepth4 = new JTextField();
jtfMeanHydraulicDepth4.setSize(80, 25);
jtfMeanHydraulicDepth4.setLocation(330, 350);
getContentPane().add(jtfMeanHydraulicDepth4);

jlblMeanHydraulicDepth4 = new JLabel("Mean Hydraulic Depth");
jlblMeanHydraulicDepth4.setSize(180, 25);
jlblMeanHydraulicDepth4.setLocation(420, 350);
getContentPane().add(jlblMeanHydraulicDepth4);

jbtnAverage = new JButton("Average");
jbtnAverage.setSize(80, 25);
jbtnAverage.setLocation(180, 440);
jbtnAverage.setMargin(new Insets(0, 0, 0, 0));
jbtnAverage.addActionListener(this);
getContentPane().add(jbtnAverage);
```

```
        jtfAverageMeanHydraulicDepth = new JTextField();
        jtfAverageMeanHydraulicDepth.setSize(80, 25);
        jtfAverageMeanHydraulicDepth.setLocation(330, 440);
        getContentPane().add(jtfAverageMeanHydraulicDepth);

        jlblAverageMeanHydraulicDepth = new JLabel("Average Mean Hydraulic Depth");
        jlblAverageMeanHydraulicDepth.setSize(180, 25);
        jlblAverageMeanHydraulicDepth.setLocation(420, 440);
        getContentPane().add(jlblAverageMeanHydraulicDepth);

        jlblComments = new JLabel("Comments:");
        jlblComments.setSize(80, 20);
        jlblComments.setLocation(30, 500);
        getContentPane().add(jlblComments);

        jtaComments = new JTextArea();
        jtaComments.setSize(450, 105);
        jtaComments.setLocation(110, 500);
        jtaComments.setBorder(new LineBorder(Color.black));
        getContentPane().add(jtaComments);

        jlblPrintingVersion = new JLabel("River Depth Estimator " + strVersion,
SwingConstants.RIGHT);
        jlblPrintingVersion.setFont(new Font(
((Font)(UIManager.get("Button.font"))).getName(), Font.ITALIC, 12));
        jlblPrintingVersion.setSize(580, 20);
        jlblPrintingVersion.setLocation(10, 660);

        oneFormatter = new DecimalFormat("0.0");


        // connector lines
        getContentPane().add(new Line(jtfBankWidth1.getLocation().x +
jtfBankWidth1.getSize().width, jtfBankWidth1.getLocation().y +
jtfBankWidth1.getSize().height/2,
                        jbtnCalculate1.getLocation().x, jtfBankWidth1.getLocation().y +
jtfBankWidth1.getSize().height/2));
        getContentPane().add(new Line(jbtnCalculate1.getLocation().x +
jbtnCalculate1.getSize().width, jbtnCalculate1.getLocation().y +
jbtnCalculate1.getSize().height/2,
                        jtfMeanHydraulicDepth1.getLocation().x,
jbtnCalculate1.getLocation().y + jbtnCalculate1.getSize().height/2));
        getContentPane().add(new Line(jbtnCalculate1.getLocation().x +
jbtnCalculate1.getSize().width/2, jbtnCalculate1.getLocation().y +
jbtnCalculate1.getSize().height,
```

```
                    jbtnCalculate1.getLocation().x + jbtnCalculate1.getSize().width/2,
jtfThalwegDepth.getLocation().y + jtfThalwegDepth.getSize().height/2));
    getContentPane().add(new Line(jbtnCalculate1.getLocation().x +
jbtnCalculate1.getSize().width/2, jtfThalwegDepth.getLocation().y +
jtfThalwegDepth.getSize().height/2,
                    jtfThalwegDepth.getLocation().x,
jtfThalwegDepth.getLocation().y + jtfThalwegDepth.getSize().height/2));

    getContentPane().add(new Line(jtfMeanderWaveLength.getLocation().x +
jtfMeanderWaveLength.getSize().width, jtfMeanderWaveLength.getLocation().y +
jtfMeanderWaveLength.getSize().height/2,
                    jbtnCalculate2.getLocation().x,
jtfMeanderWaveLength.getLocation().y + jtfMeanderWaveLength.getSize().height/2));
    getContentPane().add(new Line(jbtnCalculate2.getLocation().x +
jbtnCalculate2.getSize().width, jbtnCalculate2.getLocation().y +
jbtnCalculate2.getSize().height/2,
                    jtfMeanHydraulicDepth2.getLocation().x,
jbtnCalculate2.getLocation().y + jbtnCalculate2.getSize().height/2));

    getContentPane().add(new Line(jtfMeanderBendRadius.getLocation().x +
jtfMeanderBendRadius.getSize().width, jtfMeanderBendRadius.getLocation().y +
jtfMeanderBendRadius.getSize().height/2,
                    jbtnCalculate3.getLocation().x,
jtfMeanderBendRadius.getLocation().y + jtfMeanderBendRadius.getSize().height/2));
    getContentPane().add(new Line(jbtnCalculate3.getLocation().x +
jbtnCalculate3.getSize().width, jbtnCalculate3.getLocation().y +
jbtnCalculate3.getSize().height/2,
                    jtfMeanHydraulicDepth3.getLocation().x,
jbtnCalculate3.getLocation().y + jbtnCalculate3.getSize().height/2));

    getContentPane().add(new Line(jtfBankWidth2.getLocation().x +
jtfBankWidth2.getSize().width, jtfBankWidth2.getLocation().y +
jtfBankWidth2.getSize().height/2,
                    jbtnCalculate4.getLocation().x - 30,
jtfBankWidth2.getLocation().y + jtfBankWidth2.getSize().height/2));
    getContentPane().add(new Line(jtfSinuosity.getLocation().x +
jtfSinuosity.getSize().width, jtfSinuosity.getLocation().y +
jtfSinuosity.getSize().height/2,
                    jbtnCalculate4.getLocation().x - 30, jtfSinuosity.getLocation().y +
jtfSinuosity.getSize().height/2));
    getContentPane().add(new Line(jbtnCalculate4.getLocation().x - 30,
jtfBankWidth2.getLocation().y + jtfBankWidth2.getSize().height/2,
                    jbtnCalculate4.getLocation().x - 30, jtfSinuosity.getLocation().y +
jtfSinuosity.getSize().height/2));
    getContentPane().add(new Line(jbtnCalculate4.getLocation().x - 30,
jbtnCalculate4.getLocation().y + jbtnCalculate4.getSize().height/2,
```

```
                        jbtnCalculate4.getLocation().x, jbtnCalculate4.getLocation().y +
jbtnCalculate4.getSize().height/2));
    getContentPane().add(new Line(jbtnCalculate4.getLocation().x +
jbtnCalculate4.getSize().width, jbtnCalculate4.getLocation().y +
jbtnCalculate4.getSize().height/2,
                        jtfMeanHydraulicDepth4.getLocation().x,
jbtnCalculate4.getLocation().y + jbtnCalculate4.getSize().height/2));

    getContentPane().add(new Line(jbtnAverage.getLocation().x +
jbtnAverage.getSize().width, jbtnAverage.getLocation().y +
jbtnAverage.getSize().height/2,
                        jtfAverageMeanHydraulicDepth.getLocation().x,
jbtnAverage.getLocation().y + jbtnAverage.getSize().height/2));

    strSaveReportAsImageFolder = ".";

  }

  public void clearAverages(){
   jtfAverageMeanHydraulicDepth.setText("");
  }

  public void jmiExitClicked(){
    System.exit(0);
  }

  public void jmiPrintClicked(){
    (new PrintDialog(this)).setVisible(true);
  }

  class ImageWriterThread extends Thread{
    File outputFile;
    String strImageType;
    public ImageWriterThread(File outputFile, String strImageType){
      this.outputFile = outputFile;
      this.strImageType = strImageType;
    }
    public void run(){
      try{
        setCursor(new Cursor(Cursor.WAIT_CURSOR));
        BufferedImage panelImage = new
BufferedImage(getContentPane().getSize().width, getContentPane().getSize().height,
BufferedImage.TYPE_INT_RGB);
        Graphics2D panelImageG2D = panelImage.createGraphics();
```

```
        jlblPrintingVersion.setLocation(jlblPrintingVersion.getLocation().x,
getContentPane().getSize().height - 20);
        getContentPane().add(jlblPrintingVersion);
        getContentPane().paint(panelImageG2D);
        getContentPane().remove(jlblPrintingVersion);

        ImageIO.write(panelImage, strImageType, outputFile);
        JOptionPane.showMessageDialog(frameReference, "Report saved as " +
outputFile, "Save Successful", JOptionPane.INFORMATION_MESSAGE);
      }
      catch(Exception e){
        JOptionPane.showMessageDialog(frameReference, "Error writing file: " +
e.getMessage(),
                                      "Save Failed", JOptionPane.ERROR_MESSAGE);
      }
      setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
    }
  }

  public void jmiSaveAsClicked(){
    frameReference.requestFocus();
    JFileChooser saveReportAsFileChooser = new JFileChooser();
    saveReportAsFileChooser.setCurrentDirectory(new
File(strSaveReportAsImageFolder));
    saveReportAsFileChooser.setDialogTitle("Save As...");
    saveReportAsFileChooser.setAcceptAllFileFilterUsed(false);

    saveReportAsFileChooser.addChoosableFileFilter(new GenericFileFilter(".png",
"PNG Files"));
    saveReportAsFileChooser.addChoosableFileFilter(new GenericFileFilter(".jpg",
"JPG Files"));

saveReportAsFileChooser.setFileFilter((saveReportAsFileChooser.getChoosableFileFilte
rs())[0]);

    int approveSaveValue;
    int approveOverwriteValue = JOptionPane.NO_OPTION;
    while(! (approveOverwriteValue == JOptionPane.YES_OPTION)){
      approveSaveValue = saveReportAsFileChooser.showSaveDialog(this);
      if (approveSaveValue != JFileChooser.APPROVE_OPTION){
        return;
      }
      if ( saveReportAsFileChooser.getSelectedFile().exists() ||
          (new File(saveReportAsFileChooser.getSelectedFile().toString() +
```

```
((GenericFileFilter)(saveReportAsFileChooser.getFileFilter())).getExtension())).exists()
){  // show "replace?" dialog
        approveOverwriteValue = JOptionPane.showConfirmDialog(this, "File already
exists, overwrite?", "Confirm Overwrite",
                    JOptionPane.YES_NO_OPTION,
JOptionPane.WARNING_MESSAGE);
      }
      else{
        approveOverwriteValue = JOptionPane.YES_OPTION;
      }
    }

    strSaveReportAsImageFolder =
saveReportAsFileChooser.getCurrentDirectory().toString();
    String strSelectedFileFilterExtension =
((GenericFileFilter)(saveReportAsFileChooser.getFileFilter())).getExtension();
    String strSelectedFile = saveReportAsFileChooser.getSelectedFile().getPath();
    if (!
(strSelectedFile.toLowerCase().endsWith(strSelectedFileFilterExtension.toLowerCase()))
){
      strSelectedFile = strSelectedFile + strSelectedFileFilterExtension;
    }

    ImageWriterThread imageSaver = new ImageWriterThread(new File(strSelectedFile),
strSelectedFileFilterExtension.substring(1));
    imageSaver.start();
  }

  public void jbtnCalculate1Clicked(){
    clearAverages();
    try{
      jtfMeanHydraulicDepth1.setText(oneFormatter.format(0.12 *
Math.pow(Double.parseDouble(jtfBankWidth1.getText()), 0.69)));
      jtfThalwegDepth.setText(oneFormatter.format(0.28 *
Math.pow(Double.parseDouble(jtfBankWidth1.getText()), 0.65)));
    }
    catch(Exception e){
      JOptionPane.showMessageDialog(this, "Please input a valid number.", "Invalid
Number", JOptionPane.ERROR_MESSAGE);
    }
  }

  public void jbtnCalculate2Clicked(){
    clearAverages();
    try{
```

```
jtfMeanHydraulicDepth2.setText(oneFormatter.format(Math.exp(Math.log(Double.parse
Double(jtfMeanderWaveLength.getText()) / 240.0) / 1.52)));
    }
    catch(Exception e){
      JOptionPane.showMessageDialog(this, "Please input a valid number.", "Invalid
Number", JOptionPane.ERROR_MESSAGE);
    }

  }

  public void jbtnCalculate3Clicked(){
    clearAverages();
    try{

jtfMeanHydraulicDepth3.setText(oneFormatter.format(Math.exp(Math.log(Double.parse
Double(jtfMeanderBendRadius.getText()) / 42.0) / 1.52)));
    }
    catch(Exception e){
      JOptionPane.showMessageDialog(this, "Please input a valid number.", "Invalid
Number", JOptionPane.ERROR_MESSAGE);
    }
  }

  public void jbtnCalculate4Clicked(){
    clearAverages();
    try{
      jtfMeanHydraulicDepth4.setText(oneFormatter.format(0.09 *
Math.pow(Double.parseDouble(jtfBankWidth2.getText()), 0.59) *

Math.pow(Double.parseDouble(jtfSinuosity.getText()), 1.46)));
    }
    catch(Exception e){
      JOptionPane.showMessageDialog(this, "Please input valid numbers.", "Invalid
Number", JOptionPane.ERROR_MESSAGE);
    }
  }

  public void jbtnAverageClicked(){
    try{
      jtfAverageMeanHydraulicDepth.setText(oneFormatter.format(
(Double.parseDouble(jtfMeanHydraulicDepth1.getText()) +

Double.parseDouble(jtfMeanHydraulicDepth2.getText()) +

Double.parseDouble(jtfMeanHydraulicDepth3.getText()) +
```

```
Double.parseDouble(jtfMeanHydraulicDepth4.getText())) / 4.0 ));
    }
    catch(Exception e){
      JOptionPane.showMessageDialog(this, "All 4 values not yet calculated.", "Missing
Calculation(s)", JOptionPane.ERROR_MESSAGE);
      return;
    }
  }

  public void actionPerformed(ActionEvent ae){
    if (ae.getSource() == jmiPrint){
      jmiPrintClicked();
    }
    else if (ae.getSource() == jmiSaveAs){
      jmiSaveAsClicked();
    }
    else if (ae.getSource() == jmiExit){
      jmiExitClicked();
    }
    else if (ae.getSource() == jbtnCalculate1){
      jbtnCalculate1Clicked();
    }
    else if (ae.getSource() == jbtnCalculate2){
      jbtnCalculate2Clicked();
    }
    else if (ae.getSource() == jbtnCalculate3){
      jbtnCalculate3Clicked();
    }
    else if (ae.getSource() == jbtnCalculate4){
      jbtnCalculate4Clicked();
    }
    else if (ae.getSource() == jbtnAverage){
      jbtnAverageClicked();
    }
  }

  class Line extends JPanel{
    public Line(int startX, int startY, int endX, int endY){
      setLocation(startX, startY);
      setSize(endX - startX + 1, endY - startY + 1);
      setBackground(Color.black);
    }
  }

  public static void main(String[] args){
```

```
    RiverDepthFrame riverDepthScreen = new RiverDepthFrame();
    riverDepthScreen.setVisible(true);
  }


  public int print(Graphics g, PageFormat pf, int pageIndex)
                throws PrinterException {
    if (pageIndex >= 1) {
      return Printable.NO_SUCH_PAGE;
    }

    double pageHeight = pf.getImageableHeight();
    double pageWidth = pf.getImageableWidth();


    double widthScaleFactor = pageWidth/(double)(getContentPane().getSize().width);
    double heightScaleFactor = pageHeight/(double)(getContentPane().getSize().height);

    ((Graphics2D)g).translate(pf.getImageableX(), pf.getImageableY());
    ((Graphics2D)g).scale( Math.min(widthScaleFactor, heightScaleFactor),
Math.min(widthScaleFactor, heightScaleFactor));
    RepaintManager currentManager =
RepaintManager.currentManager(getContentPane());
    currentManager.setDoubleBufferingEnabled(false);

    getContentPane().paint(g);
    currentManager.setDoubleBufferingEnabled(true);
    return Printable.PAGE_EXISTS;
  }

  class PrintThread extends Thread{
    public void run(){
      try{
        setCursor(new Cursor(Cursor.WAIT_CURSOR));
        frameReference.requestFocus();
        PrinterJob printJob = PrinterJob.getPrinterJob();

        PageFormat pf = new PageFormat();

        printJob.setPrintable(frameReference, pf);
        printJob.setPrintService(selectedPrinter);
        PrintRequestAttributeSet printSettings = new HashPrintRequestAttributeSet(); //
set up printing attributes
        printSettings.add(new Copies(1));
        jlblPrintingVersion.setLocation(jlblPrintingVersion.getLocation().x,
getContentPane().getSize().height - 20);
```

```
            getContentPane().add(jlblPrintingVersion);
            printJob.print(printSettings);
            getContentPane().remove(jlblPrintingVersion);
          }
          catch(Exception e){
            e.printStackTrace();
          }
          setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
        }
      }


      class PrintDialog extends JDialog implements ActionListener{
        JLabel jlblPrinter;
        JComboBox jcbPrinters;
        JButton jbtnOK, jbtnCancel;
        PrintService[] availablePrinters;
        public PrintDialog(JFrame parent){
          super(parent, "Select Printer", true);
          setSize(400, 130);
          setLocation( (int)(Toolkit.getDefaultToolkit().getScreenSize().width/2.0 -
getSize().width/2.0 + 0.5),
                  (int)(Toolkit.getDefaultToolkit().getScreenSize().height/2.0 -
getSize().height/2.0 + 0.5) );
          getContentPane().setLayout(null);

          jlblPrinter = new JLabel("Printer:");
          jlblPrinter.setSize(80, 20);
          jlblPrinter.setLocation(10, 10);
          getContentPane().add(jlblPrinter);

          jcbPrinters = new JComboBox();
          jcbPrinters.setSize(380, 20);
          jcbPrinters.setLocation(10, 30);
          getContentPane().add(jcbPrinters);

          jbtnOK = new JButton("OK");
          jbtnOK.setSize(50, 25);
          jbtnOK.setLocation(140, 70);
          jbtnOK.setMargin(new Insets(0, 0, 0, 0));
          jbtnOK.addActionListener(this);
          getContentPane().add(jbtnOK);

          jbtnCancel = new JButton("Cancel");
          jbtnCancel.setSize(50, 25);
          jbtnCancel.setLocation(200, 70);
```

```
        jbtnCancel.setMargin(new Insets(0, 0, 0, 0));
        jbtnCancel.addActionListener(this);
        getContentPane().add(jbtnCancel);

        frameReference.setCursor(new Cursor(Cursor.WAIT_CURSOR));
        availablePrinters = PrinterJob.lookupPrintServices();
        frameReference.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
        Object selectedPrinter = jcbPrinters.getSelectedItem();
        jcbPrinters.removeAllItems();
        for (int i = 0; i < availablePrinters.length; i++){
          jcbPrinters.addItem(availablePrinters[i].getName());
          if ( (selectedPrinter != null) && (
((String)(selectedPrinter)).equals(availablePrinters[i].getName())) ){
            jcbPrinters.setSelectedIndex(i);
          }
        }
      }
      public void jbtnOKClicked(){
        selectedPrinter = availablePrinters[jcbPrinters.getSelectedIndex()];
        (new PrintThread()).start();
        setVisible(false);
      }
      public void jbtnCancelClicked(){
        setVisible(false);
      }
      public void actionPerformed(ActionEvent ae){
        if (ae.getSource() == jbtnOK){
          jbtnOKClicked();
        }
        else if (ae.getSource() == jbtnCancel){
          jbtnCancelClicked();
        }
      }
    }

}
```

```
package mil.navy.nrlssc.dmap.riverdepth;

import java.io.*;

public class StreamMethods{

  public static byte[] fullyReadStream(InputStream streamIn) throws Exception{
    byte[] bytesInStream = new byte[524288];  // initially 512 k
    byte[] newBytesInStream;
    byte[] buffer = new byte[65536];
    int bytesReadThisPass = 0;
    int totalBytesRead = 0;

    while ( (bytesReadThisPass = streamIn.read(buffer, 0, buffer.length)) != -1){
      while ( (bytesReadThisPass + totalBytesRead) > bytesInStream.length ){
        newBytesInStream = new byte[bytesInStream.length + 524288];
        System.arraycopy(bytesInStream, 0, newBytesInStream, 0, totalBytesRead);
        bytesInStream = newBytesInStream;
      }
      System.arraycopy(buffer, 0, bytesInStream, totalBytesRead, bytesReadThisPass);
      totalBytesRead = totalBytesRead + bytesReadThisPass;
    }

    newBytesInStream = new byte[totalBytesRead];
    System.arraycopy(bytesInStream, 0, newBytesInStream, 0, totalBytesRead);
    return newBytesInStream;
  }

}
```

```
package mil.navy.nrlssc.dmap.riverdepth;

import java.io.File;
import javax.swing.*;
import javax.swing.filechooser.*;

public class GenericFileFilter extends FileFilter{

  String strExtension, strDescription;

  public GenericFileFilter(String strExtension, String strDescription){
    this.strExtension = strExtension;
    this.strDescription = strDescription;
  }

  public boolean accept(File fileToCheck){
    if (fileToCheck.isDirectory()) {
      return true;
    }
    if (fileToCheck.getName().toLowerCase().endsWith(strExtension.toLowerCase())){
      return true;
    }
    return false;
  }

  public String getExtension(){
    return strExtension;
  }

  public String getDescription(){
    return strDescription + " (*" + strExtension + ")";
  }

}
```

```
package mil.navy.nrlssc.dmap.riverdepth;

import java.awt.*;
import java.io.*;
import javax.swing.*;

public class Utility{

  public static Image getImageFromJar(String strImagePathInJar){
    Image loadedImage = null;
    try{
      InputStream imageStream =
Thread.currentThread().getContextClassLoader().getResourceAsStream(strImagePathInJ
ar);

      loadedImage =
Toolkit.getDefaultToolkit().createImage(StreamMethods.fullyReadStream(imageStream)
);

      imageStream.close();

      MediaTracker mt = new MediaTracker(new JPanel());
      mt.addImage(loadedImage, 0);
      mt.waitForAll();
    }
    catch(Exception e){
      e.printStackTrace();
    }
    return loadedImage;
  }

}
```